Computer Organization and Software Systems

CONTACT SESSION 3

BITS Pilani
Pilani Campus

Pruthvi Kumar K R

Today's Session

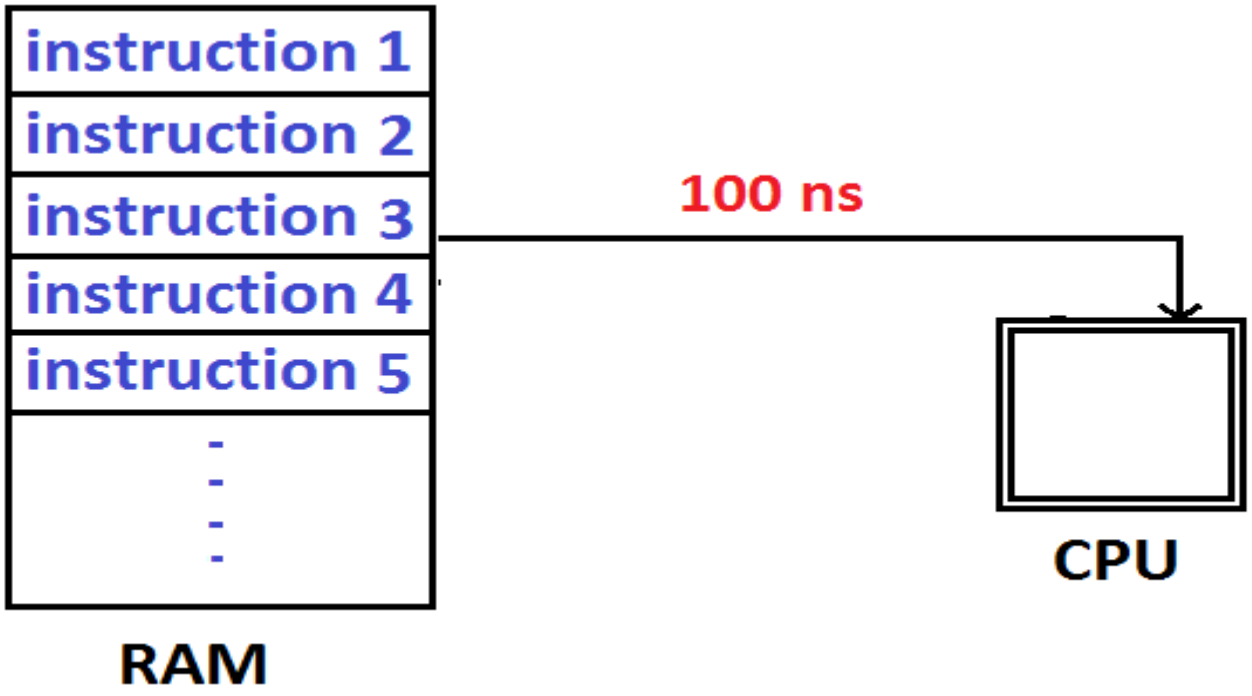
Contact Hour	List of Topic Title	Text/Ref Book/external resource

5-6	• Memory Hierarchy
	• Locality
	  • Locality of Reference to Program Data
	  • Locality of instruction fetches
	• Cache Memories
	  • Generic Cache Memory Organization
	  • Direct-Mapped Caches
	  • Fully Associative Caches
		T1

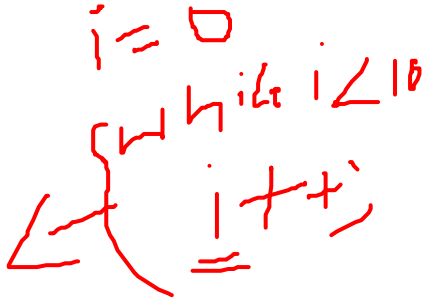
Memory Hierarchy

- Registers
  - In CPU
- Internal or Main memory
  - May include one or more levels of cache
  - "RAM"
- External memory
  - Backing store

Performance enhancement - Motivation



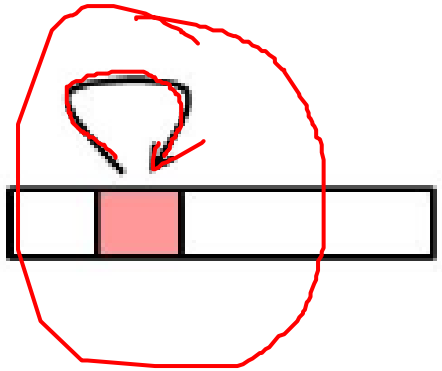
Performance enhancement - Motivation

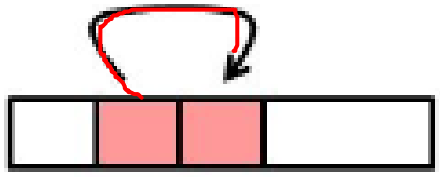


Locality of Reference

During the course of the execution of a program, memory references tend to cluster

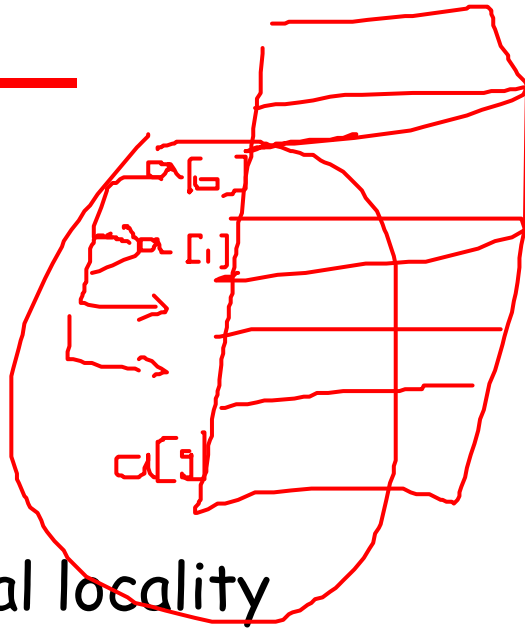
- **Temporal locality:** Locality in time
  - If an item is referenced, it will tend to be referenced again soon
- **Spatial locality:** Locality in space
  - If an item is referenced, items whose addresses are close by will tend to be referenced soon.





Example

```
product  = 1;
for ( i = 0; i < n-1; i++)
        product = product *  a[i] ;
```



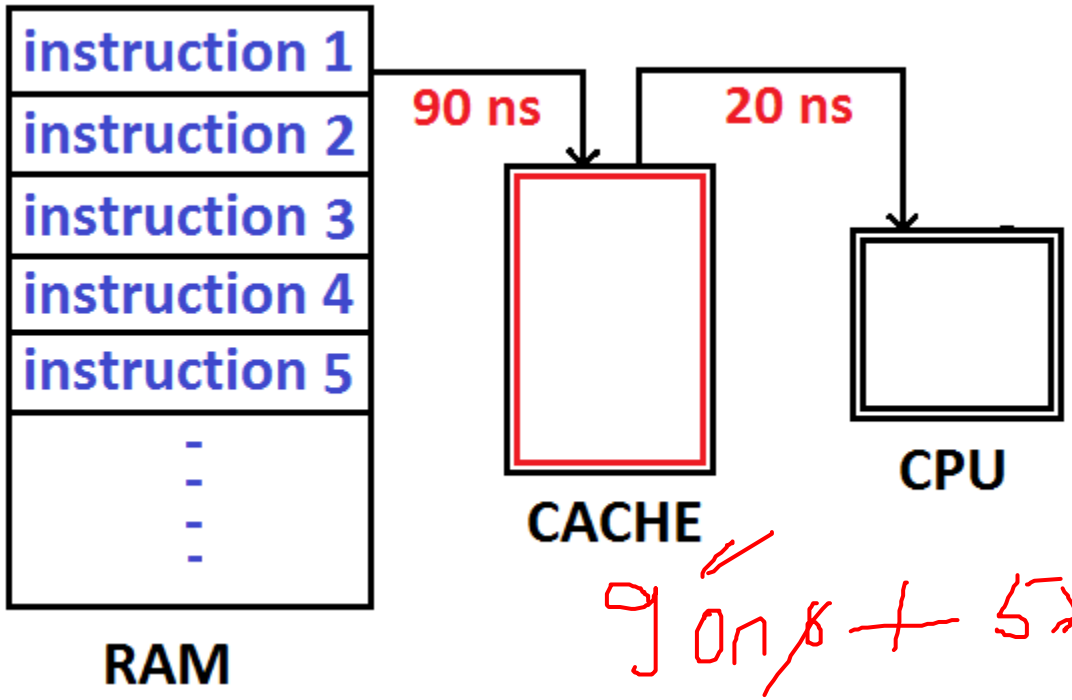
Data :

- Access array elements in succession – spatial locality
- Reference to "product" in each iteration – Temporal locality

Instructions :

- Reference instructions in sequence : Spatial locality
- Looping through : Temporal locality

Performance enhancement - Motivation



90ns + 5×20

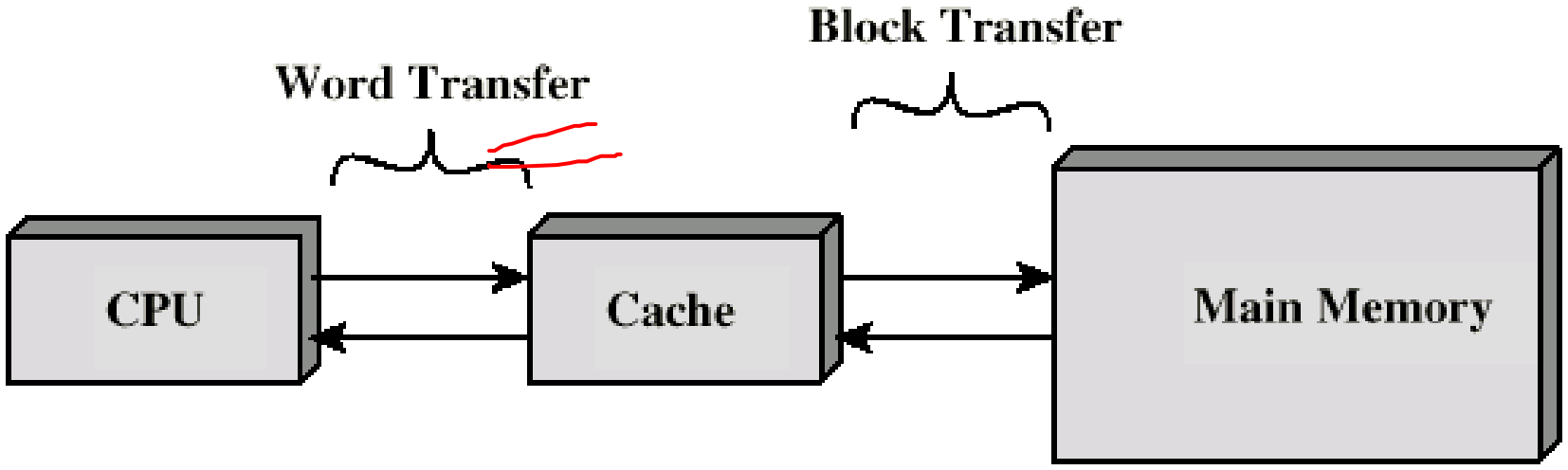
=> 190ns

Cache

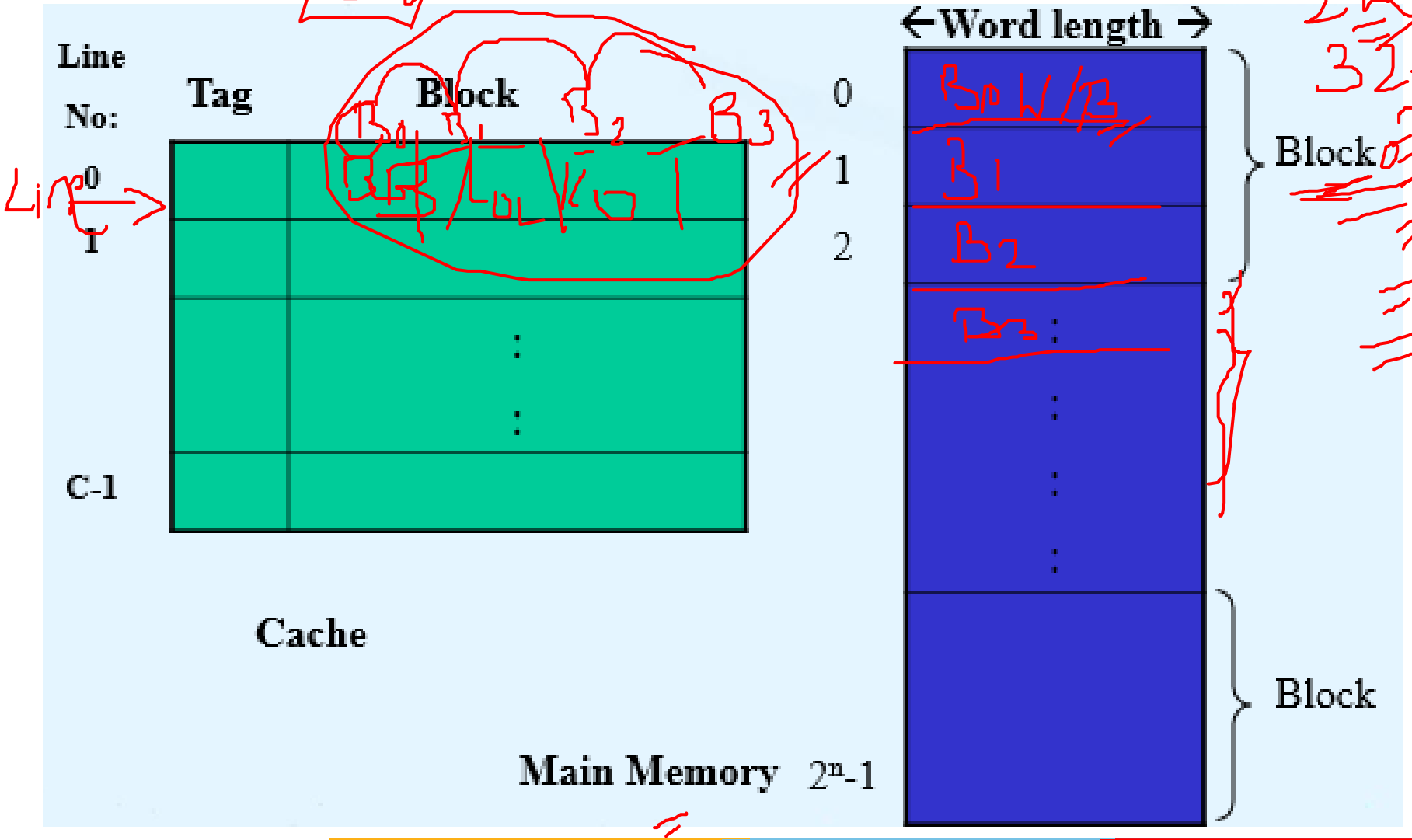
BITS Pilani
Pilani Campus

Cache

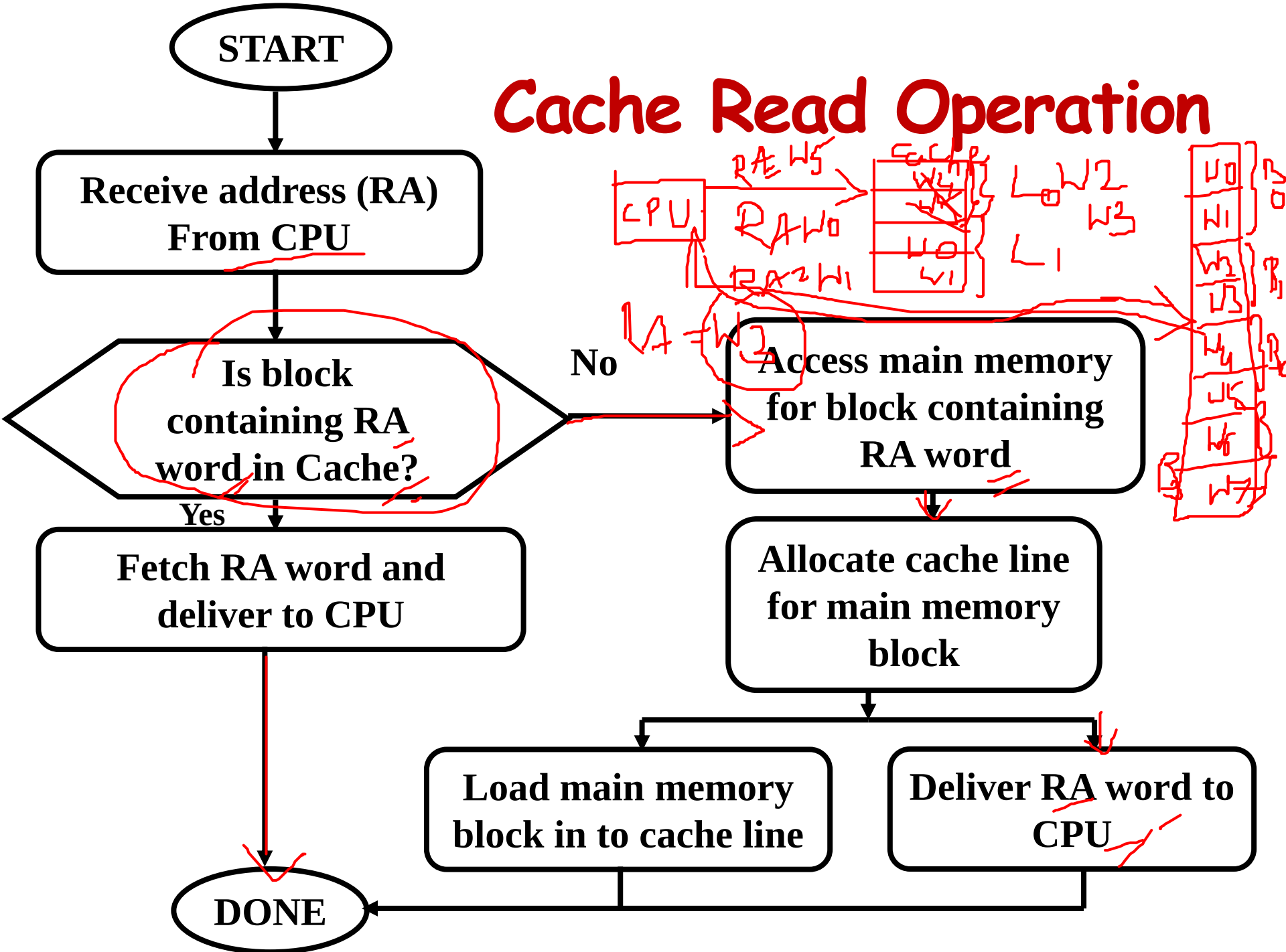
- Small, fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or separate module



Cache and Main Memory Structure



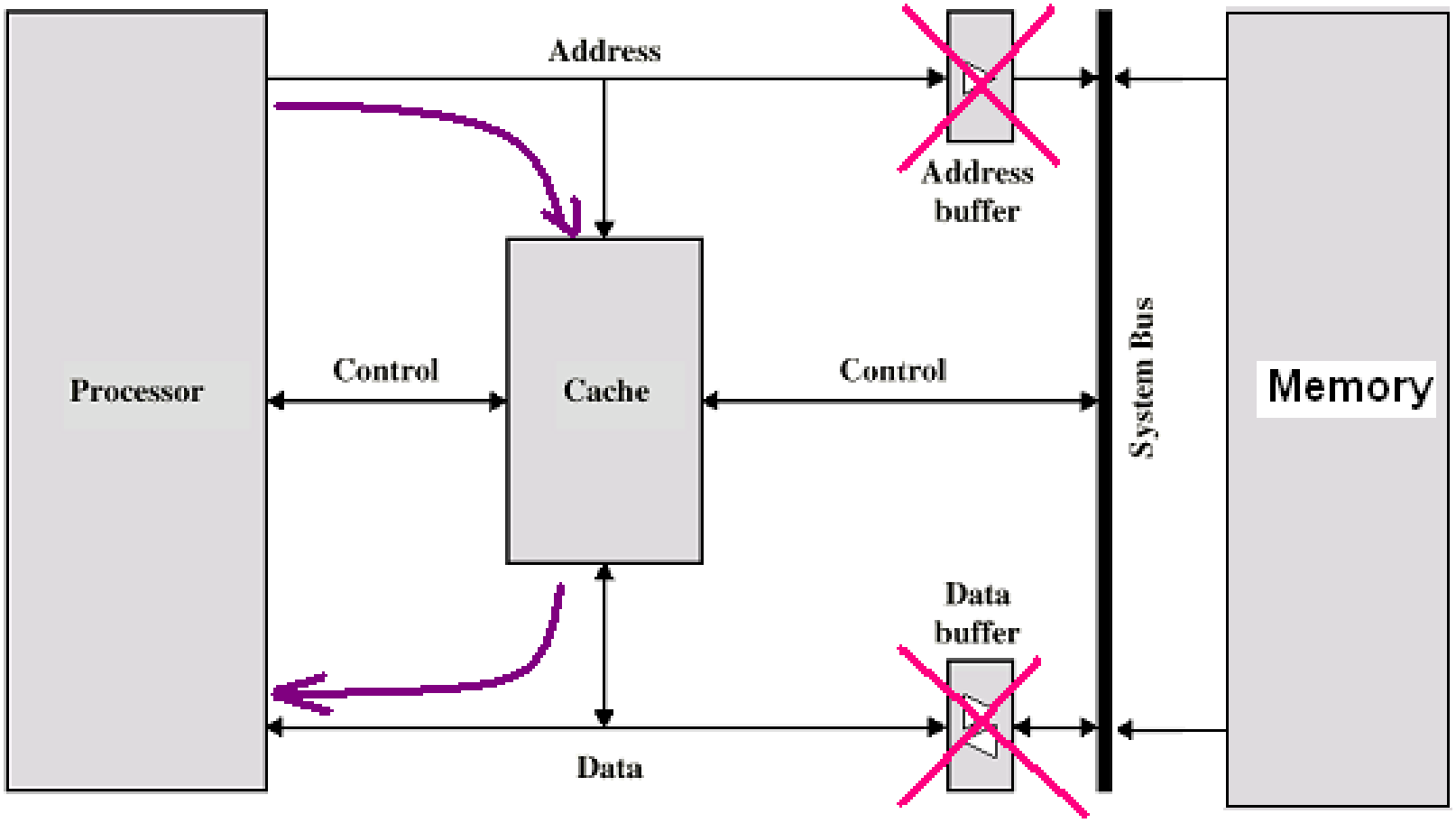
Cache Read Operation



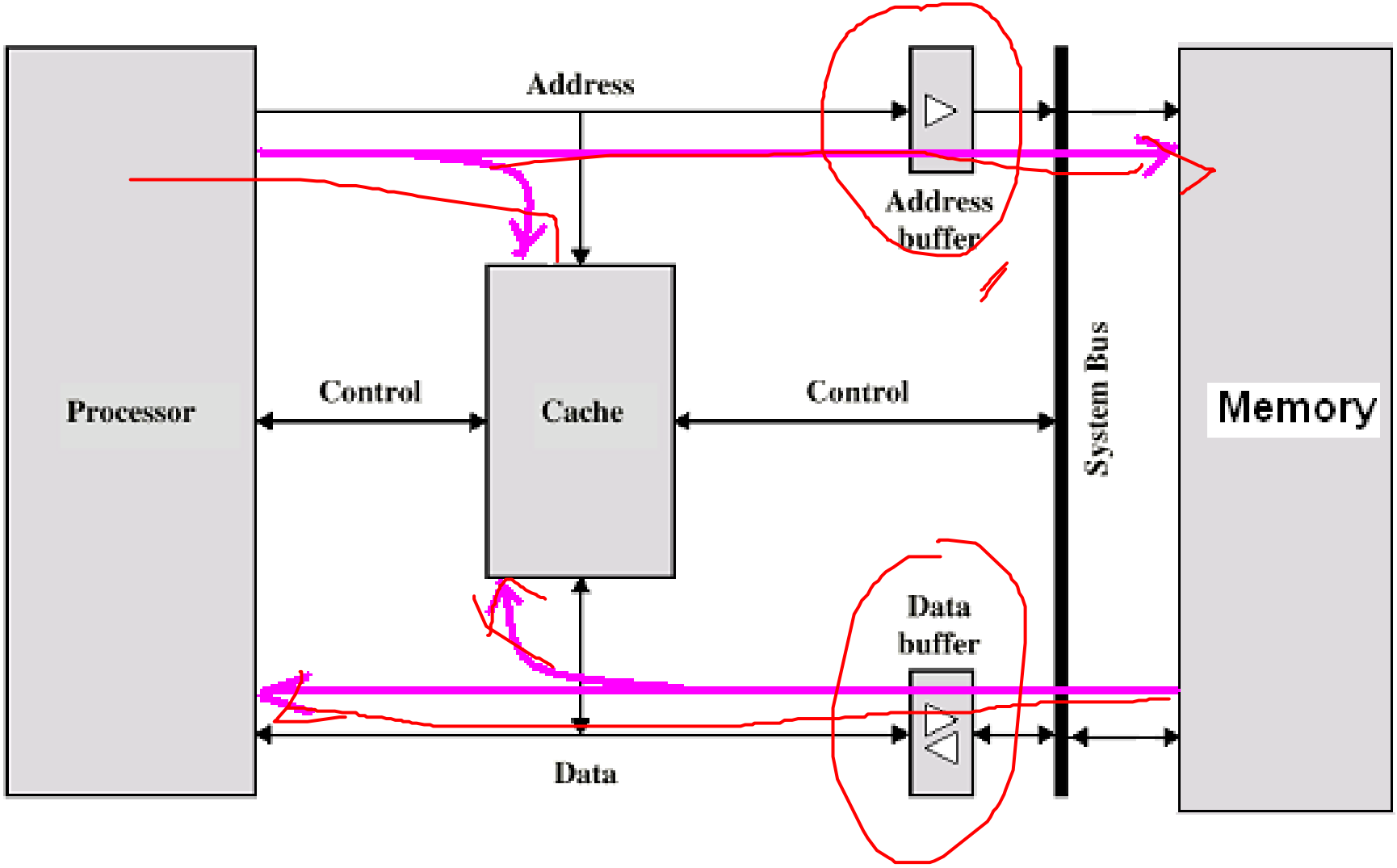
Performance of cache

- Hit ratio : Number of Hits / total references to memory
- Hit
- Miss

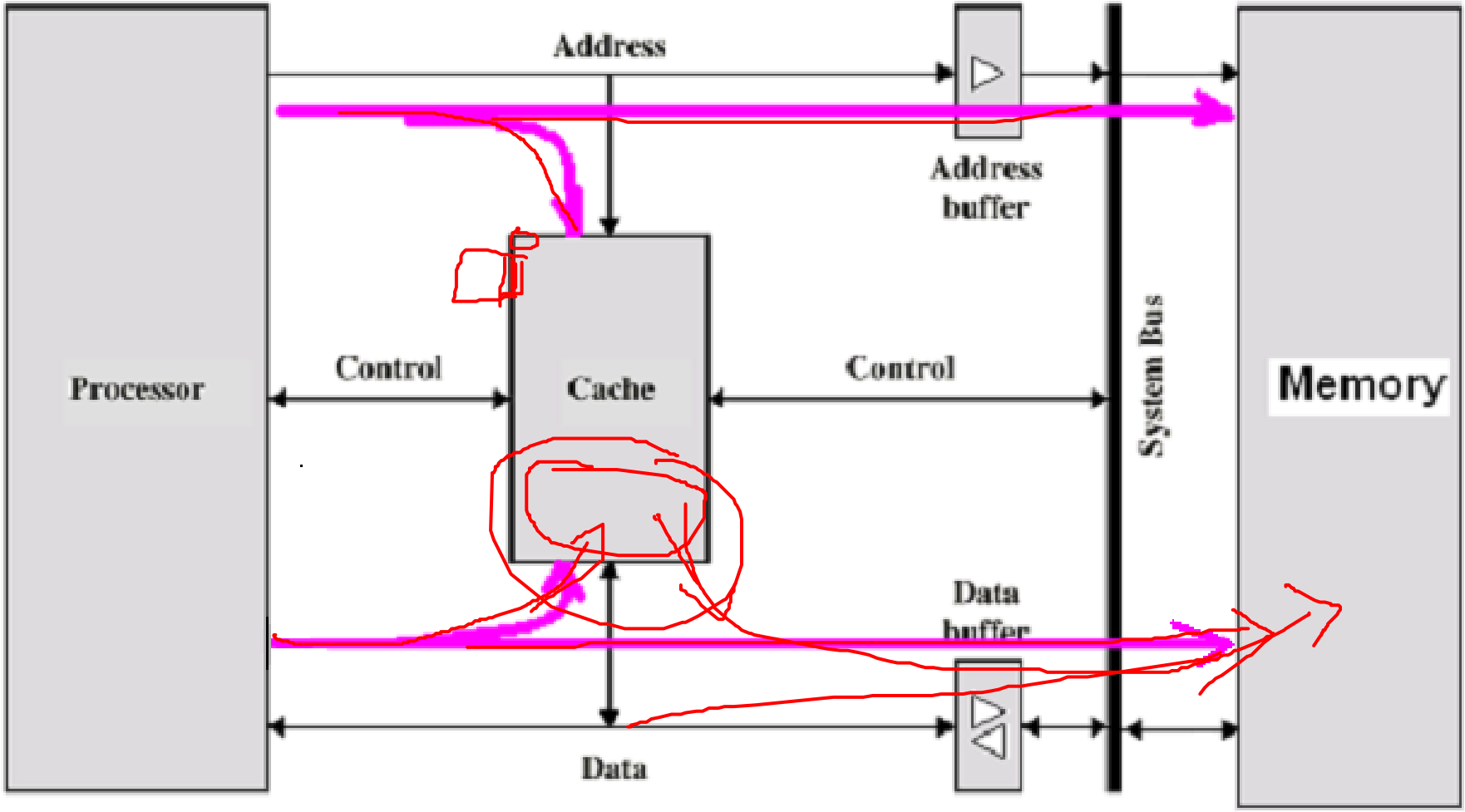
Read Hit


Address
Address buffer
Processor
Control
Cache
Control
System Bus
Memory
Data buffer
Data

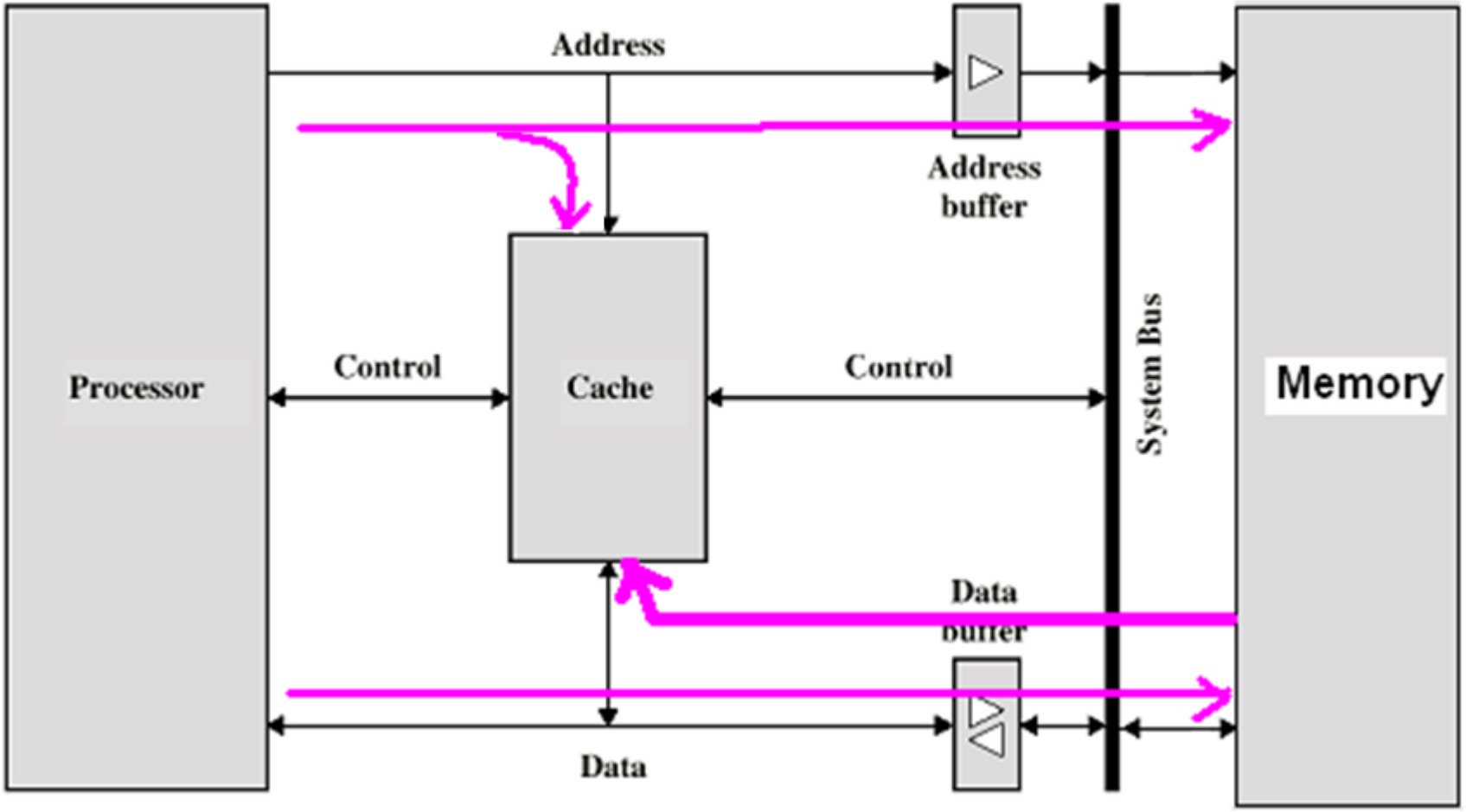
Read Miss


Address
Address buffer
Processor
Control
Cache
Control
System Bus
Memory
Data buffer
Data

Write hit


Address
Address buffer
Processor
Control
Cache
Control
System Bus
Memory
Data buffer
Data

Write miss

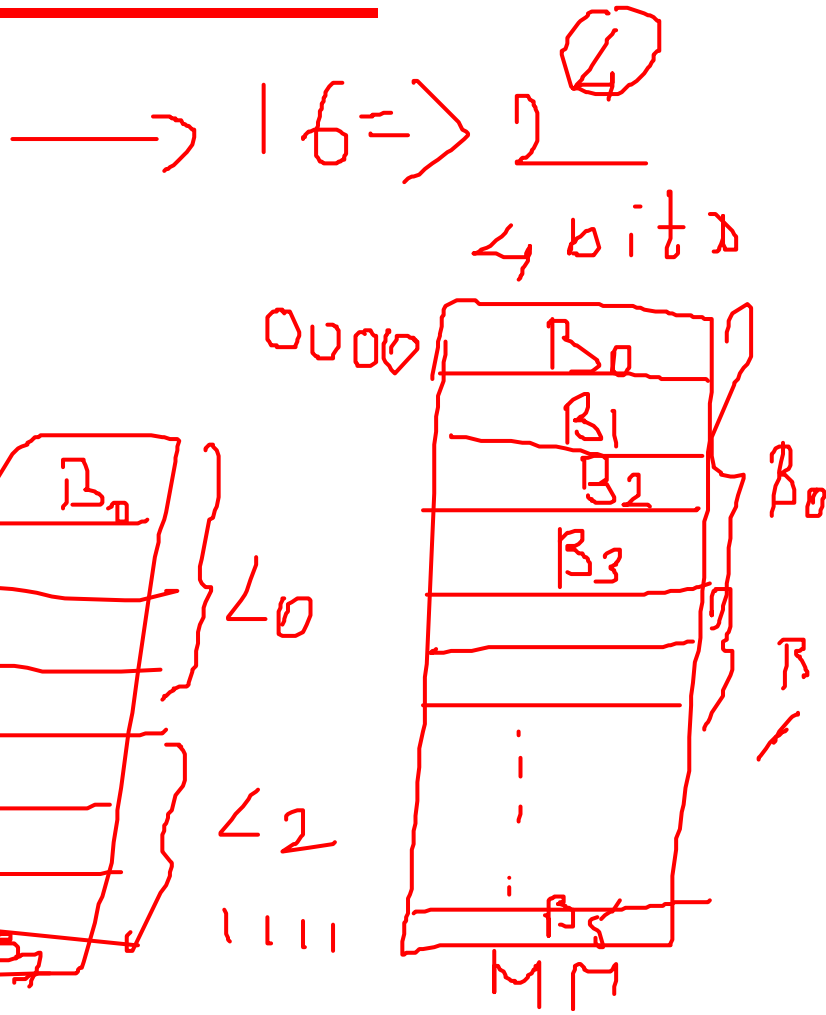

Address
Address buffer
Processor
Control
Cache
Control
System Bus
Memory
Data buffer
Data

Mapping Function

- How memory blocks are mapped to cache lines
- Three types
  - Direct mapping
  - Associative mapping
  - Set Associative mapping

Mapping Function

- 16 Bytes main memory
  - How many address bits are required?
- Memory block size is 4 bytes
- Cache of 8 Byte
  - How many cache lines?



Mapping Function

- 16 Bytes main memory
  - How many address bits are required?
- Memory block size is 4 bytes
- Cache of 8 Byte
  - How many cache lines?

4 bits

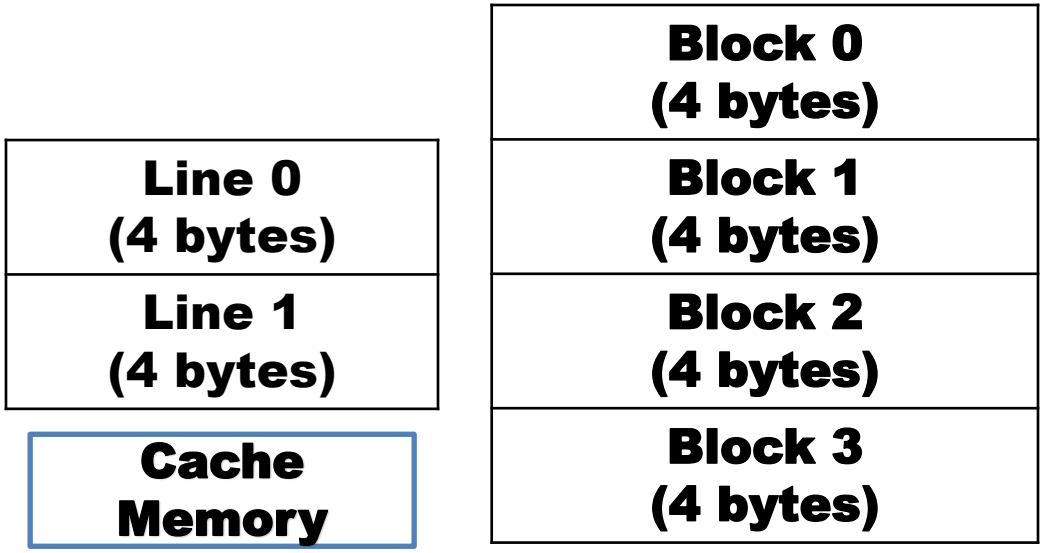
Mapping Function

- 16 Bytes main memory
  - How many address bits are required?
- Memory block size is 4 bytes
- Cache of 8 Byte
  - How many cache lines?

cache is 2 lines (4 bytes per Line)

Mapping Function

- 16 Bytes main memory
  - How many address bits are required?
- Memory block size is 4 bytes
- Cache of 8 Byte
  - How many cache lines?
  - cache is 2 lines (4 bytes per Line)



Cache Size = No of Lines * Line Size

MM Size = No of Blocks * Line Size

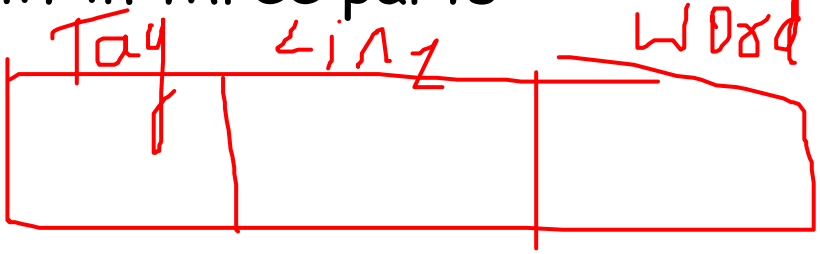
Direct Mapping

- Each block of main memory maps to only one cache line
  - **i.e. if a block is in cache, it must be in one specific place**
  - **i = j modulo m**

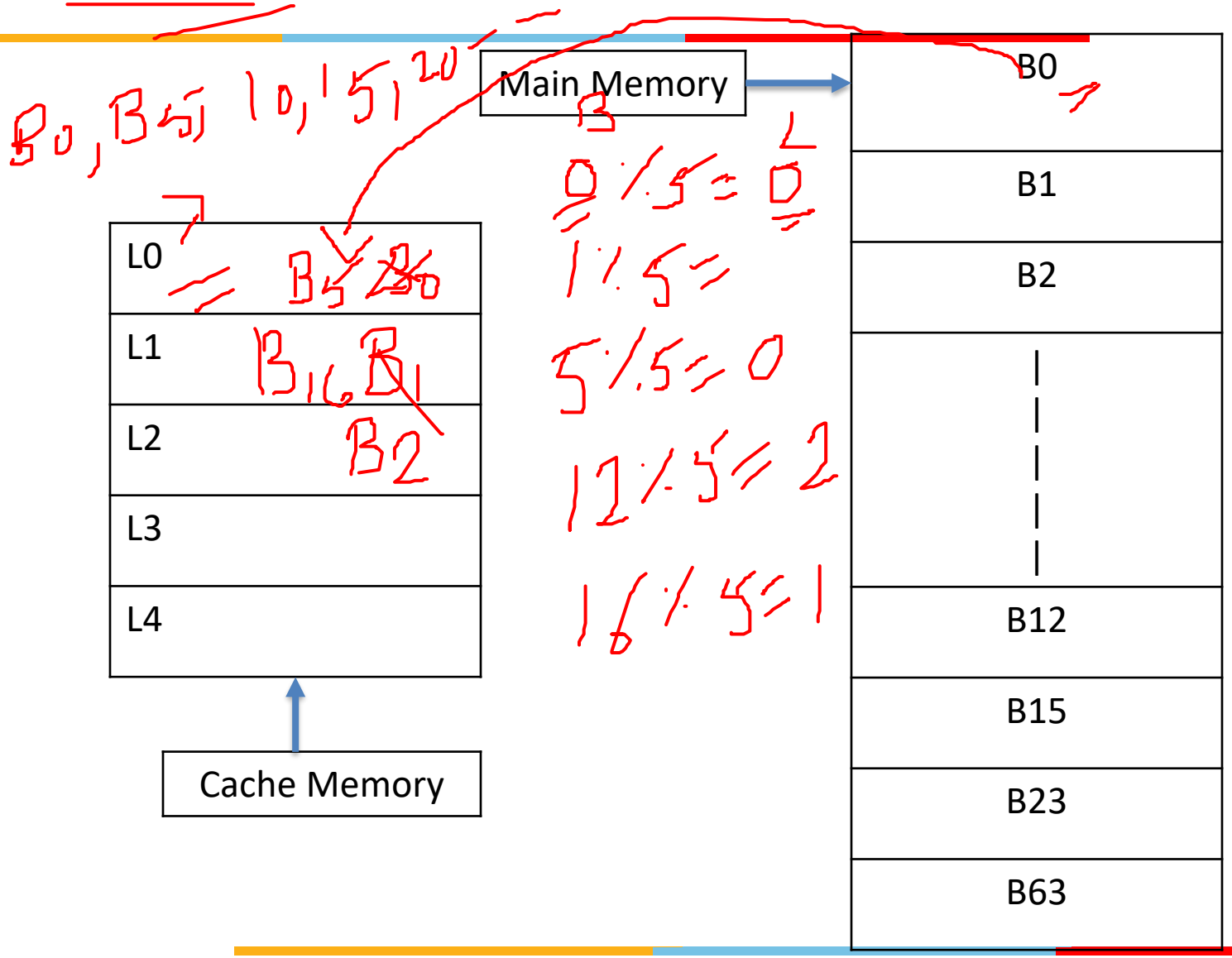
  where i = cache line number

  j = main memory block no.

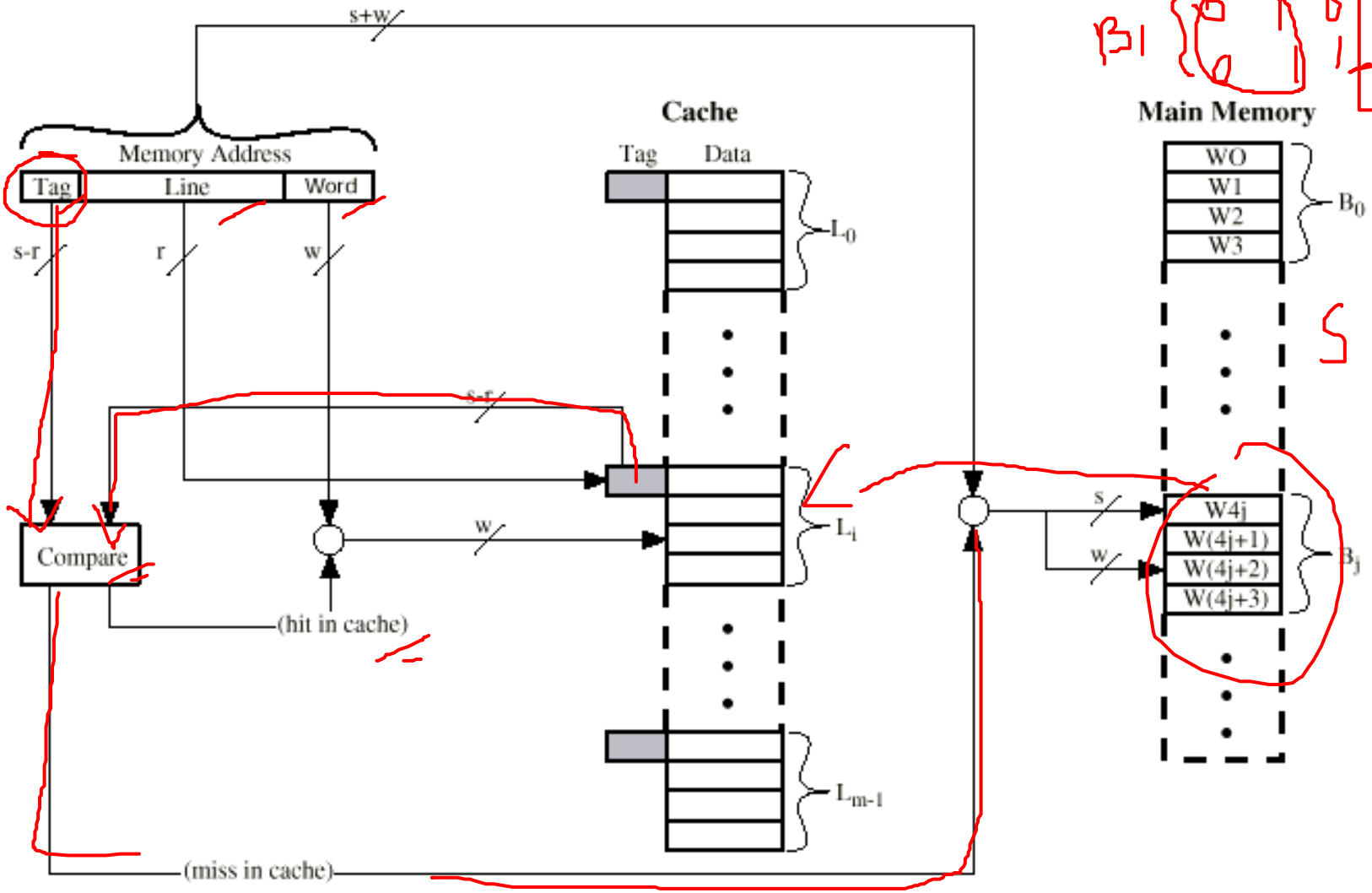
  m = no.of lines in the cache
- Address is split in three parts:
  - Tag
  - Line
  - Word



Direct Mapping Cache Organization



Direct Mapping Cache Organization



Direct mapping- Summary

Address length = (s+w) bits

Number of addressable units = 2^{s+w} words or bytes

Block size = line size = 2^{w} words or bytes

Number of blocks in main memory = $2^{s+w} / 2^{w} = 2^{s}$

Number of lines in cache = m = 2^{r}

Size of tag =(s-r) bits

Direct Mapping pros & cons

- Simple
- Inexpensive
- Fixed location for given block
  - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

Problem 1

1k = 2^{10}
1M = 2^{20}
1G = 2^{30}

Given :

- Cache of 64kByte, Cache block of 4 bytes
- 16MBytes main memory ⇒ 16 * 1M => $2^4 * 2^{20} = 2^{24}$

Find out

a) Number of bits required to address the memory
b) Number of blocks in main memory = $\frac{2^{24}}{2^2} = 2^{22}$
c) Number of cache lines
d) Number of bits required to identify a word (byte) in a block?
e) Number of bits to identify a block
f) Tag, Line, Word

Solution

Given :

- Cache of 64kByte, Cache block of 4 bytes
- 16MBytes main memory

Find out

a) Number of bits required to address the memory

24 bits

b) Number of blocks in main memory

4M blocks

c) Number of cache lines

16K lines

Solution 1

innovate achieve lead

Given :

- **Cache of 64kByte, Cache block of 4 bytes**
- **16MBytes main memory**

Find out

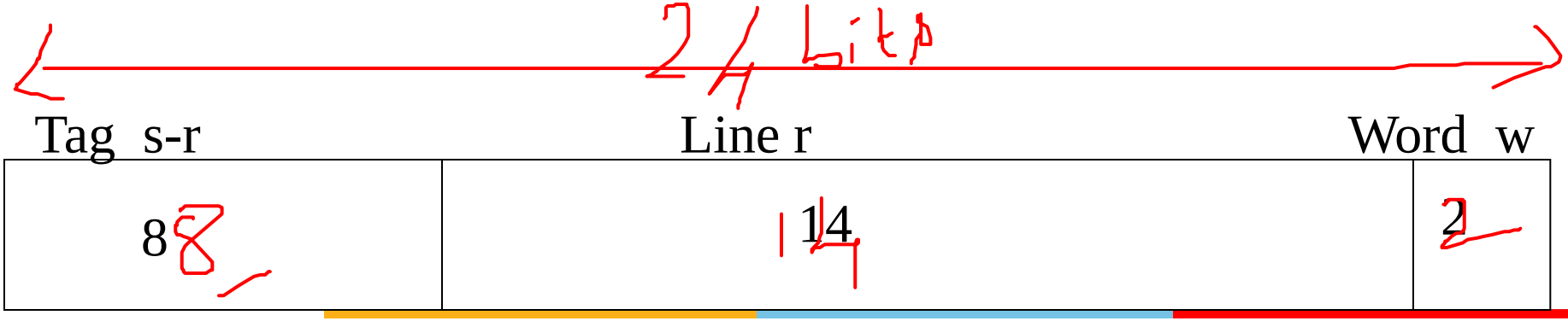
d) Number of bits required to identify a word (byte) in a block?

2 bits

e) Number of bits required to identify block

22 bits

f) Tag, Line, Word



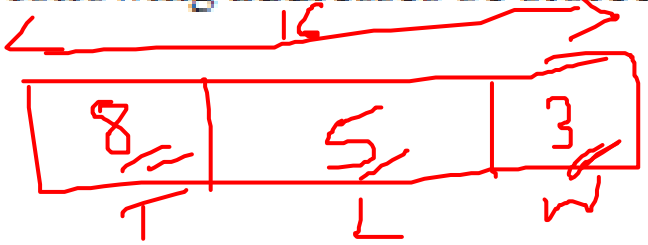
Problem 2

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

a. How is a 16-bit memory address divided into tag, line number, and byte number?

b. Into what line would bytes with each of the following addresses be stored?

0001	0001	0001	1011
1100	0011	0011	0100
1101	0000	0001	1101
1010	1010	1010	1010



c. Suppose the byte with address 0001 1010 0001 1010 is stored in the cache. What are the addresses of the other bytes stored along with it?

d. How many total bytes of memory can be stored in the cache?

e. Why is the tag also stored in the cache?

Solution 2

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

a. How is a 16-bit memory address divided into tag, line number, and byte number?

TAG =8	LINE=5	WORD=3

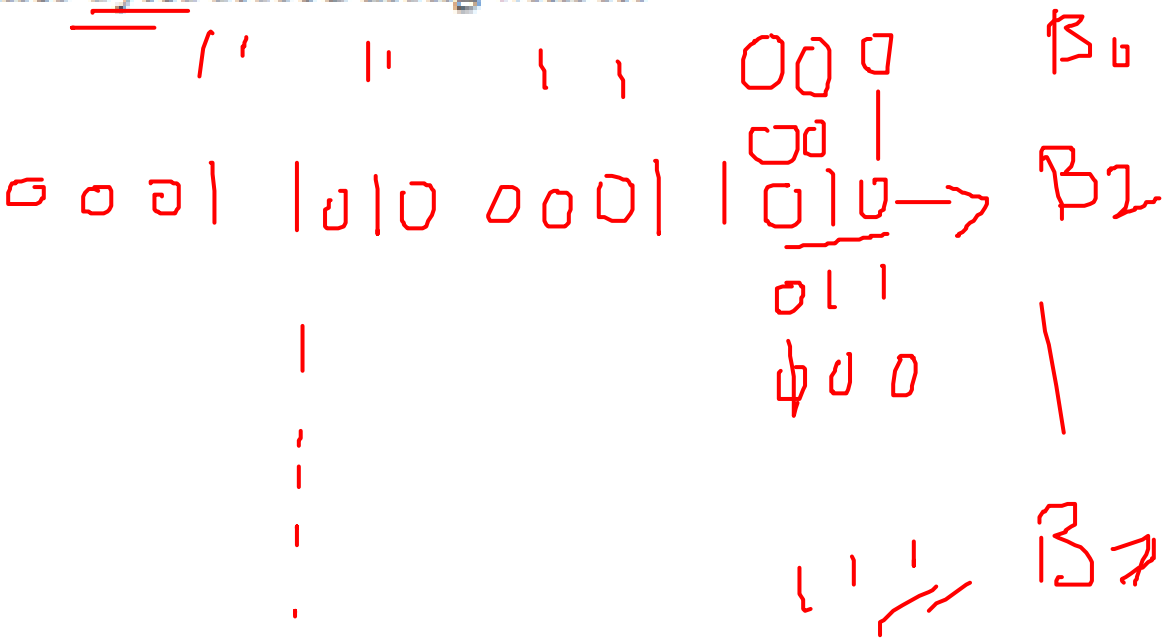
b. Into what line would bytes with each of the following addresses be stored?

0001	0001	0001	1011	→ 23
1100	0011	0011	0100	→ 24
1101	0000	0001	1101	→ 29
1010	1010	1010	1010	→ 21

Solution 2

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

c. Suppose the byte with address 0001 1010 0001 1010 is stored in the cache. What are the addresses of the other bytes stored along with it?



Solution 2

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

d. How many total bytes of memory can be stored in the cache?

: Number of cache line 32

Block size : 8 bytes

Total bytes saved in cache = 32 x 8 bytes = 256bytes (excluding tag)

Tag bits saved : 32 x 8 bits = 256 bits = 32 bytes

Total bytes saved in cache = 256bytes + 32 bytes = 288 Bytes (including tag)

Solution 2

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

e. Why is the tag also stored in the cache?

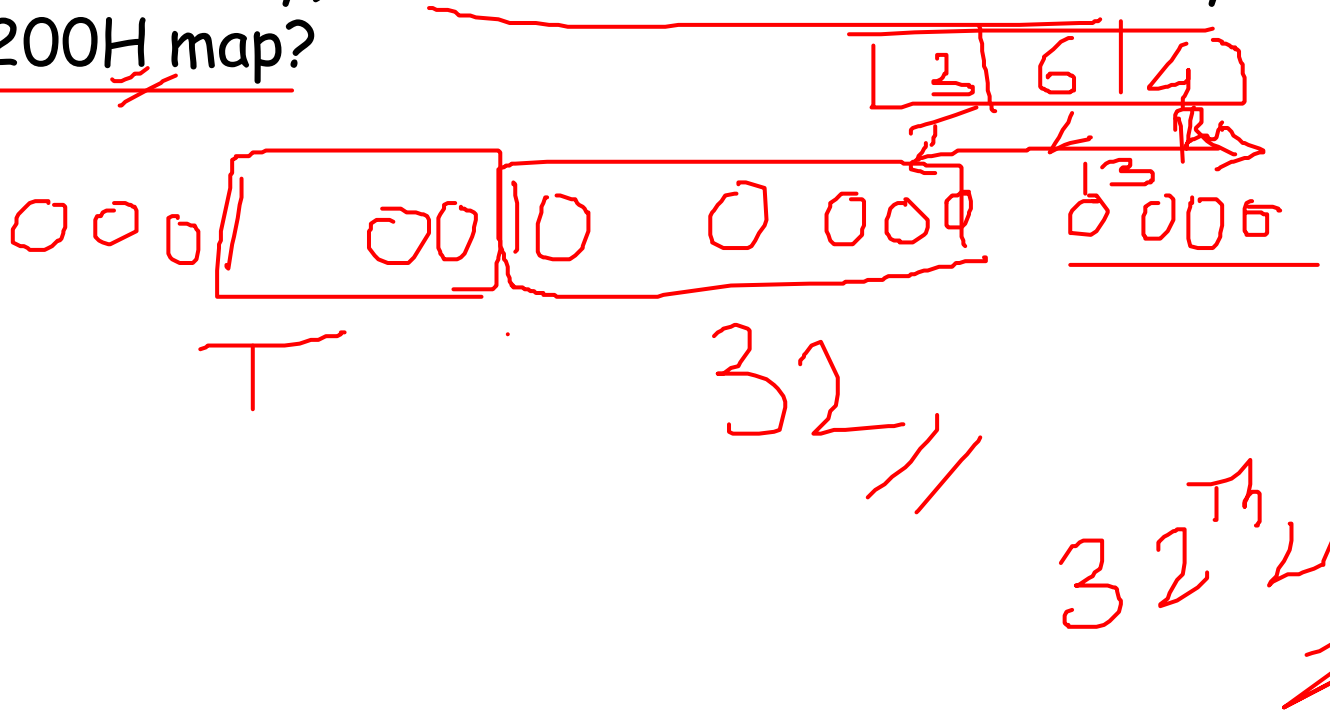
Two or more blocks can be mapped to same cache line.

To distinguish between the blocks, tag bits are used.

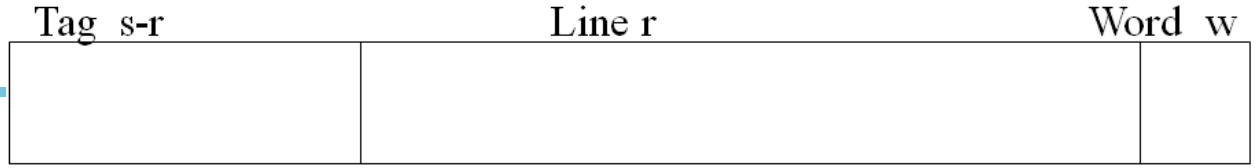
Problem 3

Consider a direct-mapped cache with 64 cache lines and a block size of 16 bytes and main memory of 8K (Byte addressable memory) . To what line number does byte address 1200H map?

32nd line.



Problem 4



The system uses a L1 cache with direct mapping and 32-bit address format is as follows:

bits 0 - 3 = offset (word)

bits 4 - 14 = index bits (Line)

bits15 - 31 = tag

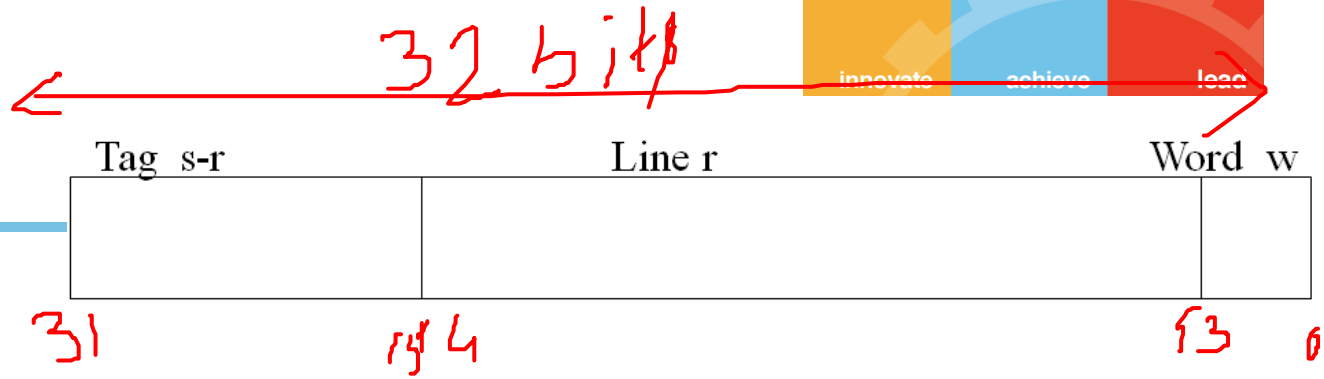
a) What is the size of cache line?

b) How many Cache lines are there?

c) How much space is required to store the tags for the L1 cache?

d) What is the total Capacity of cache including tag storage?

Problem 4



The system uses a L1 cache with direct mapping and 32-bit address format is as follows:

bits 0 - 3 = offset (word)

bits 4 - 14 = index bits (Line)

bits15 - 31 = tag

a) What is the size of cache line?

b) How many Cache lines are there?

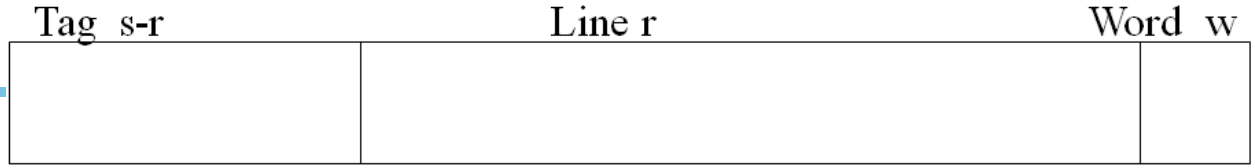
c) How much space is required to store the tags for the L1 cache?

d) What is the total Capacity of cache including tag storage?

a) Size of cache line

= size of the block = 2^4 = 16 bytes

Problem 4



The system uses a L1 cache with direct mapping and 32-bit address format is as follows:

bits 0 - 3 = offset (word)

bits 4 - 14 = index bits (Line)

bits15 - 31 = tag

a) What is the size of cache line?

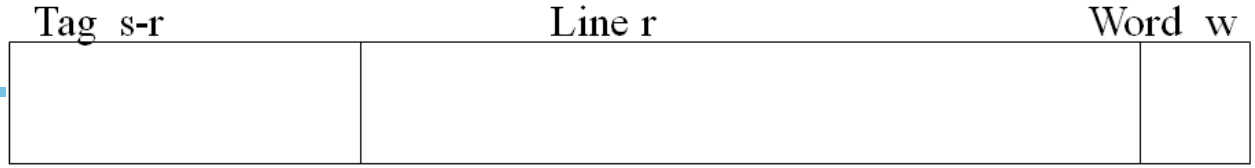
b) How many Cache lines are there?

c) How much space is required to store the tags for the L1 cache?

d) What is the total Capacity of cache including tag storage?

b) No. of cache lines = 2^11= 2K cache lines

Problem 4



The system uses a L1 cache with direct mapping and 32-bit address format is as follows:

bits 0 - 3 = offset (word)

bits 4 - 14 = index bits (Line)

bits15 - 31 = tag

a) What is the size of cache line?

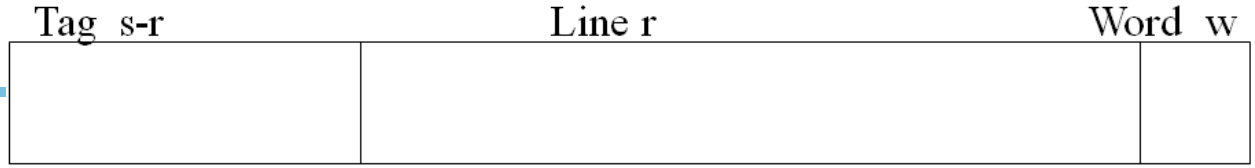
b) How many Cache lines are there?

c) How much space is required to store the tags for the L1 cache?

d) What is the total Capacity of cache including tag storage?

Space for tag = No. cache lines * Tag length = 2K * 17 bits =34 K bits

Problem 4



The system uses a L1 cache with direct mapping and 32-bit address format is as follows:

bits 0 - 3 = offset (word)

bits 4 - 14 = index bits (Line)

bits15 - 31 = tag

a) What is the size of cache line?

b) How many Cache lines are there?

c) How much space is required to store the tags for the L1 cache?

d) What is the total Capacity of cache including tag storage?

Total capacity = 34Kbits + 32 Kbytes

Problem 5

- 16 Bytes main memory, Memory block size is 4 bytes, Cache of 8 Byte (cache is 2 lines of 4 bytes each)
- Block access sequence :

0 2 0 2 2 0 0 2 0 0 0 2 1

M M H M H M H H M M H H H M

- Find out hit ratio.

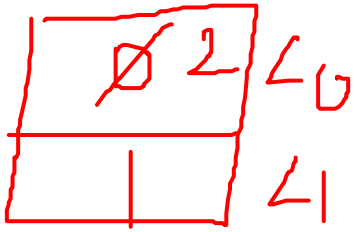
4/13

0 % 2 = 0

2 % 2 = 0

0 % 2 = 0

1 % 2



Problem 5

- 16 Bytes main memory, Memory block size is 4 bytes, Cache of 8 Byte (cache is 2 lines of 4 bytes each)
- Block access sequence :

0 2 0 2 2 0 0 2 0 0 0 2 1

Block 0 will be placed in line 0 0%2 ➔ 0

Block 1 will be placed in line 1 1%2 ➔ 1

Block 2 will be placed in line 2 2%2 ➔ 0

- Find out hit ratio.

Problem 5

- 16 Bytes main memory, Memory block size is 4 bytes, Cache of 8 Byte (cache is 2 lines of 4 bytes each)
- Block access sequence :

0 2 0 2 2 0 0 2 0 0 0 2 1

- Find out hit ratio.

Problem 5

- 16 Bytes main memory, Memory block size is 4 bytes, Cache of 8 Byte (cache is 2 lines of 4 bytes each)
- Block access sequence :

0 2 0 2 2 0 0 2 0 0 0 2 1

- Find out hit ratio.

4/13=30.76%

Problem 6

Suppose a 1024-byte cache has an access time of 0.1 microseconds and the main memory stores 1 Mbytes with an access time of 1 microsecond. A referenced memory block that is not in cache must be loaded into cache .

Answer the following questions:

a) What is the number of bits needed to address the main memory?

b) If the cache hit ratio is 95%, what is the average access time for a memory reference?

Problem 6

Suppose a 1024-byte cache has an access time of 0.1 microseconds and the main memory stores 1 Mbytes with an access time of 1 microsecond. A referenced memory block that is not in cache must be loaded into cache .

Answer the following questions:

a) What is the number of bits needed to address the main memory?

20 bits

a) If the cache hit ratio is 95%, what is the average access time for a memory reference?

Solution 6

b) If the cache hit ratio is 95%, what is the average access time for a memory reference?

Avg access time = hit ratio * cache access +

(1- hit ratio) * (cache access + memory access)

= .95 * 0.1 microsec + .05 * (1 + 0.1) microsec

= .095 + .055 microsec

= 0.15 microseconds